Tight Binding Model for Bravais Lattice

Jason Ts
jasonphysics19@gmail.com

June 28, 2020

Contents

Chapter 1

A Brief to Tight Binding Model

Tight binding model is a well-known method to solve the energy band of lattice in condensed matter physics. By second quantization, energy-momentum(of electrons)($E_{\vec{k}} - \vec{k}$) could be solved.

1.1 Bravais/Non-Bravais lattice

Crystal is constructed by lattices, which could translate along the primitive vectors $\{\vec{R}_i\}$ to fill the crystal. Atoms in non-Bravais lattice could be completely got by

$$\sum_{h,k,l}\left(h\vec{R}_1 + k\vec{R}_2 + l\vec{R}_3\right) + \sum_j \vec{r}_j$$

In Bravais lattice,

$$\sum_{h,k,l}\left(h\vec{R}_1 + k\vec{R}_2 + l\vec{R}_3\right)$$

In Bravais lattice of 3D case, there are 7 crystal systems and 14 kinds of lattices.

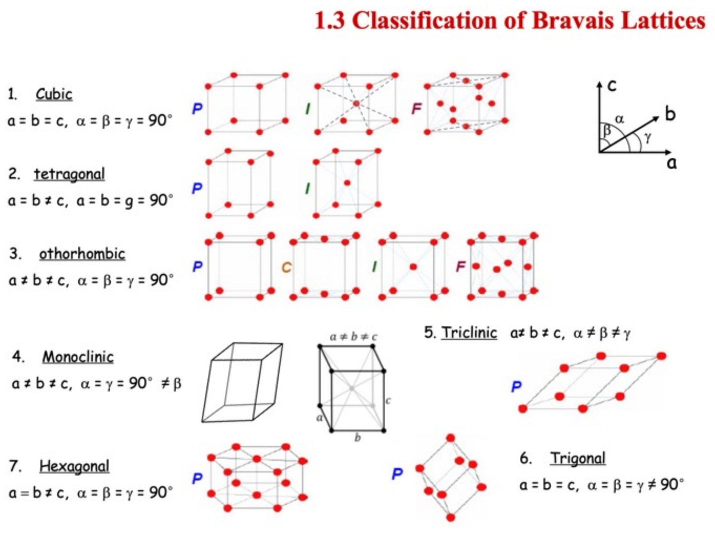

1.3 Classification of Bravais Lattices
1. Cubic
a = b = c, α = β = γ = 90°
P
I
F
c
b
a
α
β
γ
2. tetragonal
a = b ≠ c, a = b = g = 90°
P
I
3. othorhombic
a ≠ b ≠ c, α = β = γ = 90°
P
C
I
F
4. Monoclinic
a ≠ b ≠ c, α = γ = 90° ≠ β
a ≠ b ≠ c
5. Triclinic a≠ b ≠ c, α ≠ β ≠ γ
P
6. Trigonal
a = b = c, α = β = γ ≠ 90°
P
7. Hexagonal
a = b ≠ c, α = β = γ = 90°
P

Figure 1.1: 3D Bravais lattice

1.2 Tight Binding Model-Second Quantization

1.2.1 Fermi particle

Plane wave	$\lvert\vec{k}\rangle$	$\frac{1}{\sqrt{V}}e^{i\vec{k}\cdot\vec{r}}$
Bloch wave	$\lvert n\vec{k}\rangle$	$e^{i\vec{k}\cdot\vec{r}}u_{\vec{k}(\vec{r})}$
Wannier function	$\lvert n_j\rangle$	$a_{nj}(\vec{r}-\vec{R})$

1.2.2 Creation/Annihilation operator

vacuum state $\lvert 0\rangle$	
creation operator	$a_j^\dagger$
annihilation operator	a_j

They satisfy

$$\begin{cases} a_j^\dagger|0\rangle = |j\rangle & a_j^\dagger|1_j\rangle = 0 \quad a_j^\dagger a_j^\dagger|0\rangle = 0 \\ (a_j)^2 = 0 & \left(a_j^\dagger\right)^2 = 0 \\ \begin{cases} \left[a_i, a_j^\dagger\right]_+ = \delta_{i,j} \\ [a_i, a_j]_+ = \left[a_i^\dagger, a_j^\dagger\right]_+ = 0 \end{cases} \end{cases}$$

To generalize, use $n_j(=0,1)$ to illustrate the number of atoms on state $|n_j\rangle$.

$$|\cdots, n_{j-1}, n_j, n_{j+1}, \cdots\rangle = \prod_j (a_j^\dagger)^{n_j}|0\rangle$$

1.2.3 Exchange Energy

Wannier function is $|\phi(\vec{r})$. Exchange energy between two atoms is

$$-J_{m,n} = \langle\phi(\vec{R}_m)|\Delta V|\phi(\vec{R}_n)\rangle$$

So the Halmitonian could be written as

$$\hat{H} = \sum_{h,k,l,j,\vec{k}} \varepsilon_{h,k,l,j,\vec{k}} C^\dagger_{h,k,l,j,\vec{k}} C_{h,k,l,j,\vec{k}}$$

In it $N_{h,k,l,j,\vec{k}} = C^\dagger_{h,k,l,j,\vec{k}} C_{h,k,l,j,\vec{k}}$ is the number of particles (0/1) on the orbit of (h,k,l,j), $C^\dagger_{h,k,l,j,\vec{k}}/C_{h,k,l,j,\vec{k}}$ are creation/annihilation operators.
In tight binding model, the Halmitonian is

$$H = -t\sum_j \left(C_j^\dagger C_j + C_{j+1}^\dagger C_j\right) - t'\sum_j \left(C_j^\dagger C_{j+2} + C_{j+2}^\dagger C_j\right)$$

It is just a simplified form in 1D case in Bravais lattice, there is not enough space for the 3D case. In tight binding model only the item with t would be considered.

1.2.4 Fourier Transform

$$\begin{cases} c^\dagger_{h,k,l,j,\vec{k}} = \sum_{\vec{k}} e^{-i\vec{k}\left(h\vec{R}_1+k\vec{R}_2+l\vec{R}_3+\vec{r}_j\right)} c^\dagger_{\vec{k}} \frac{1}{\sqrt{V}} \\ c_{h,k,l,j,\vec{k}} = \sum_{\vec{k}} e^{i\vec{k}\left(h\vec{R}_1+k\vec{R}_2+l\vec{R}_3+\vec{r}_j\right)} c_{\vec{k}} \frac{1}{\sqrt{V}} \end{cases}$$

(still 1D case in Bravais lattice)

$$\begin{aligned} H &= -t_1 \sum_j \sum_k \sum_{k'} \left[e^{-ikja} c_k^\dagger e^{ik'(j+1)a} c_{k'} + e^{ik'(j+1)a} c_{k'}^\dagger e^{ikja} c_k \right] \frac{1}{V} \\ &= -t_1 \sum_k c_k^\dagger c_k (e^{ika} + e^{-ika}) \end{aligned}$$

Chapter 2

Contents of the Source Code

The whole codes are written in python, including 8 files, in which 7 files are 7 crystal systems and Lattice3D.py is the main file to collect the class defined in each .py file. Every single crystal system .py file could run independently. However, it is recommended to use Lattice3D.py since it has the same use as every single .py has and more functions are available such as plotting and refining.
This project is based on **numpy**, **scipy**, **matplotlib** and **copy**. Be sure to install those packages before use the file.
Source codes had been uploaded on Github

Chapter 3

How to use

3.1 Conventions

3.1.1 Lattice Definition

The name of lattice's type refers to the picture below For example, the first kind of Cubic will be named as

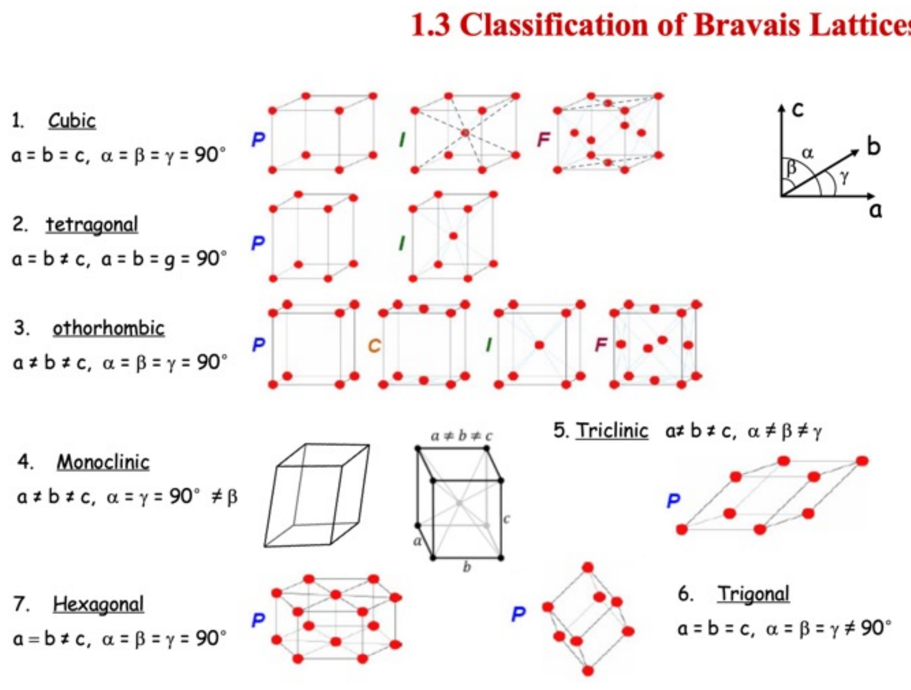
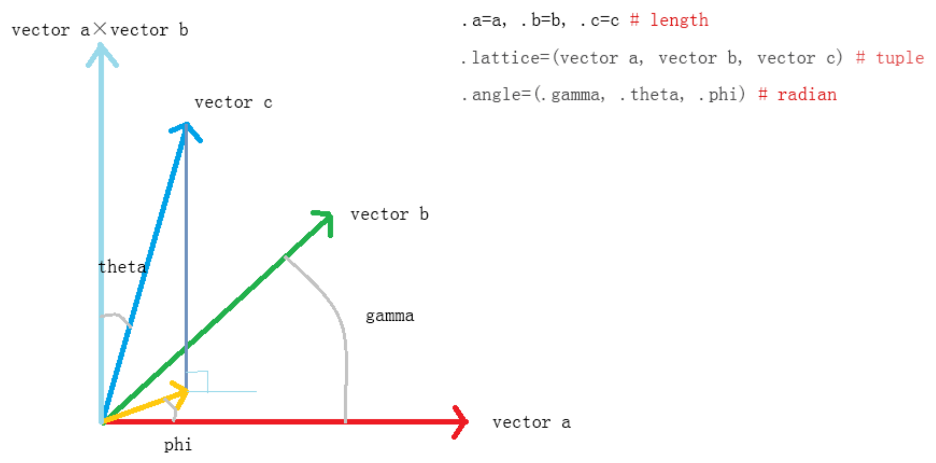

1.3 Classification of Bravais Lattices
1. Cubic
a = b = c, α = β = γ = 90°
P
I
F
c
b
a
α
β
γ
2. tetragonal
a = b ≠ c, a = b = g = 90°
P
I
3. othorhombic
a ≠ b ≠ c, α = β = γ = 90°
P
C
I
F
4. Monoclinic
a ≠ b ≠ c, α = γ = 90° ≠ β
a ≠ b ≠ c
5. Triclinic a≠ b ≠ c, α ≠ β ≠ γ
P
7. Hexagonal
a = b ≠ c, α = β = γ = 90°
P
P
6. Trigonal
a = b = c, α = β = γ ≠ 90°

Figure 3.1: 3D Bravais lattice

Cubic_1, Hexagonal is named as *Hexagonal*. Pay attention to the underline. The code follows the lengths and angles convention as illustrate in the graph.

3.1.2 Vector Convention



```
.a=a, .b=b, .c=c # length
.lattice=(vector a, vector b, vector c) # tuple
.angle=(.gamma, .theta, .phi) # radian
```

Figure 3.2: Vector Convention

Suppose the edges of lattice are defined by 3 vectors above: $\vec{a}$, $\vec{b}$ and $\vec{c}$ (Hexagonal's angle between $\vec{a}$ and $\vec{b}$ is 120°). $\vec{a}$ and $\vec{b}$ is in the $x-y$ plane, and $\vec{a}$ is parallel to the x axis. *alpha*, *beta* and *gamma* are angles between $\vec{a}$, $\vec{b}$ and $\vec{c}$, as shown on the top right corner of . With the length of $\vec{a}$, $\vec{b}$ and $\vec{c}$ and the angle *gamma*, *theta* and *phi* can determine the shape of lattice.

3.2 How to use the class Lattice

3.2.1 class value

Since the main concern on the tight binding model is that when the type and the parameter of the lattice are known, what a picture is like in the lattice, the input of Lattice (class) in Lattice3D.py like
lat=Lattice(lattice='Triclinic',t=1,a=(1,2,3),angle=(60,30,45))
String input in the variable *lattice* is in the type of lattice. t is the exchange energy. a is the lengths of $\vec{a}$, $\vec{b}$ and $\vec{c}$. 3 angles in the variable *angle* are *gamma*, *theta* and *phi*.

3.2.2 Properties of class Lattice

property	usage	property	usage
lat.t	exchange energy	lat.phi	angle between $\vec{a}$ and the projection of $\vec{c}$ on plane $\vec{a}$, $\vec{b}$
lat.a	length of $\vec{a}$	lat.angle	(lat.gamma,lat.theta,lat.phi)
lat.b	length of $\vec{b}$	lat.lattice	3 lattice edges' coordinates
lat.c	length of $\vec{c}$	lat.reciprocal	3 reciprocal vectors
lat.alpha	angle between $\vec{a}$ and $\vec{c}$	lat.normal	tuple of $\vec{a}\times\vec{b}$, $\vec{b}\times\vec{c}$ and $\vec{c}\times\vec{a}$
lat.beta	angle between $\vec{b}$ and $\vec{c}$	lat.V	volume of 3 primitive vectors
lat.gamma	angle between $\vec{a}$ and $\vec{b}$	lat.lattice_V	volume of the lattice
lat.angle_lattice	(alpha,beta,gamma)	lat.reciprocal_V	volume of 3 reciprocal vectors
lat.vector	3 primitive vectors	lat.order	will be explained in chapter 4
lat.theta	angle between $\vec{c}$ and $\vec{a}\times\vec{b}$	lat.energy(kx,ky,kz)	energy at (k_x, k_y, k_z)

3.2.3 Additions

1. You can also input the angle in radian type,
   lat=Lattice((np.pi/3,np.pi/6,np.pi/4),radian_type=True)

2. Since error in computer calculation could not be avoided, variable err is set to measure the equality of length. If $|a - b| < err$, then regard $a = b$.
   *lat=Lattice(err=10**-8)*

Chapter 4

Features

4.1 n-th nearest atoms

Lattice can not only search for the nearest atoms, by counting the variable t's length (which means t input is a tuple/list) n, it will also search for the n-th nearest atoms.
$t\_reverse$ is a variable to show whether sort/reverse the t or not. Default None, which means do not sort.
$lat.order$(leaves in page 8) is a tuple. For elements in it, it contains 4 parts. the first is the length between atoms, the second contains many angles (theta,phi) to determine the direction, third is the coordinate on respect of the second part's angles, the fourth part is the t.

4.2 Plot energy-momentum graph

Plotting the graph of energy-momentum ($E - \vec{k}$). Take $Plot2D$ as an example, if variable $k = (None, 0, 0)$, it plots the graph of $E - k_x$ when $k_y = 0$ and $k_z = 0$.
Plot2D(lat.energy,k=(None,0,0))

4.3 Plot Wigner Seitz unit cell

Plot the 3D graph of WS cell. Input the class(lat)/primitive vector(lat.vector). n is the density of points to search, it will affects how the graph's perspective changing smoothly and how the planes of cell close at the edge. $refine\_vec$: find the shortest primitive vectors or not(recommended).
Plot_WS_cell(lat.vector,n=100)

4.4 Plot Lattice

Plotting the 3D graph of lattice. Input the class(lat)/primitive vector(lat.vector). $layer$ is how many cells to be plotted ($(2 \times layer)^3$).
PlotLattice(lat.vector,layer=1)